
Some Tools to Help With Cluster and Grid Computing

**Self Adapting Numerical Algorithms
(SANS) Effort,
LAPACK for Clusters, and
PAPI**

**Jack Dongarra
Innovative Computing Laboratory
University of Tennessee**

Overview

- Self Adapting Numerical Algorithms (SANS) Effort
- LAPACK for Clusters
- PAPI
- Work sponsored by ...



Scientific Discovery through
Advanced Computing (SciDAC)



Next Generation Software (NGS)



Motivation Self Adapting Numerical Software (SANS) Effort

- Optimizing software to exploit the features of a given processor has historically been an exercise in hand customization.
 - Time consuming and tedious
 - Hard to predict performance from source code
 - Growing list of kernels to tune
 - Must be redone for every architecture and compiler
 - Compiler technology often lags architecture
 - Best algorithm may depend on input, so some tuning may be needed at run-time.
 - Not all algorithms semantically or mathematically equivalent
 - Need for quick/dynamic deployment of optimized routines.

What is Self Adapting Performance Tuning of Software?

- Two steps:
 1. Identify and generate a space of algorithm/software, with various
 - Instruction mixes and orders
 - Memory Access Patterns
 - Data structures
 - Mathematical Formulations
 2. Search for the fastest one, by running them
- When do we search?
 - Once per kernel and architecture
 - At compile time
 - At run time
 - All of the above
- Many examples
 - PhiPAC, ATLAS, Sparsity, FFTW, Spiral,...

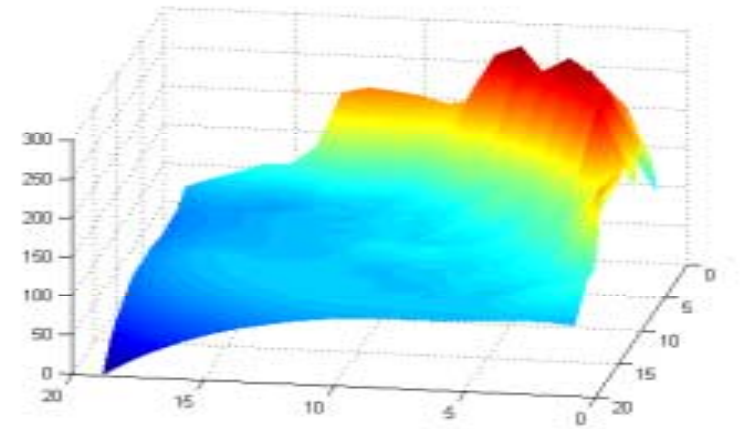
Self Adapting Numerical Software - SANS Effort

- Provide software technology to aid in high performance on commodity processors, clusters, and grids.
- Pre-run time (library building stage) and run time optimization.
- Integrated performance modeling and analysis
- Automatic algorithm selection – polyalgorithmic functions
- Automated installation process
- Can be expanded to areas such as communication software and selection of numerical algorithms



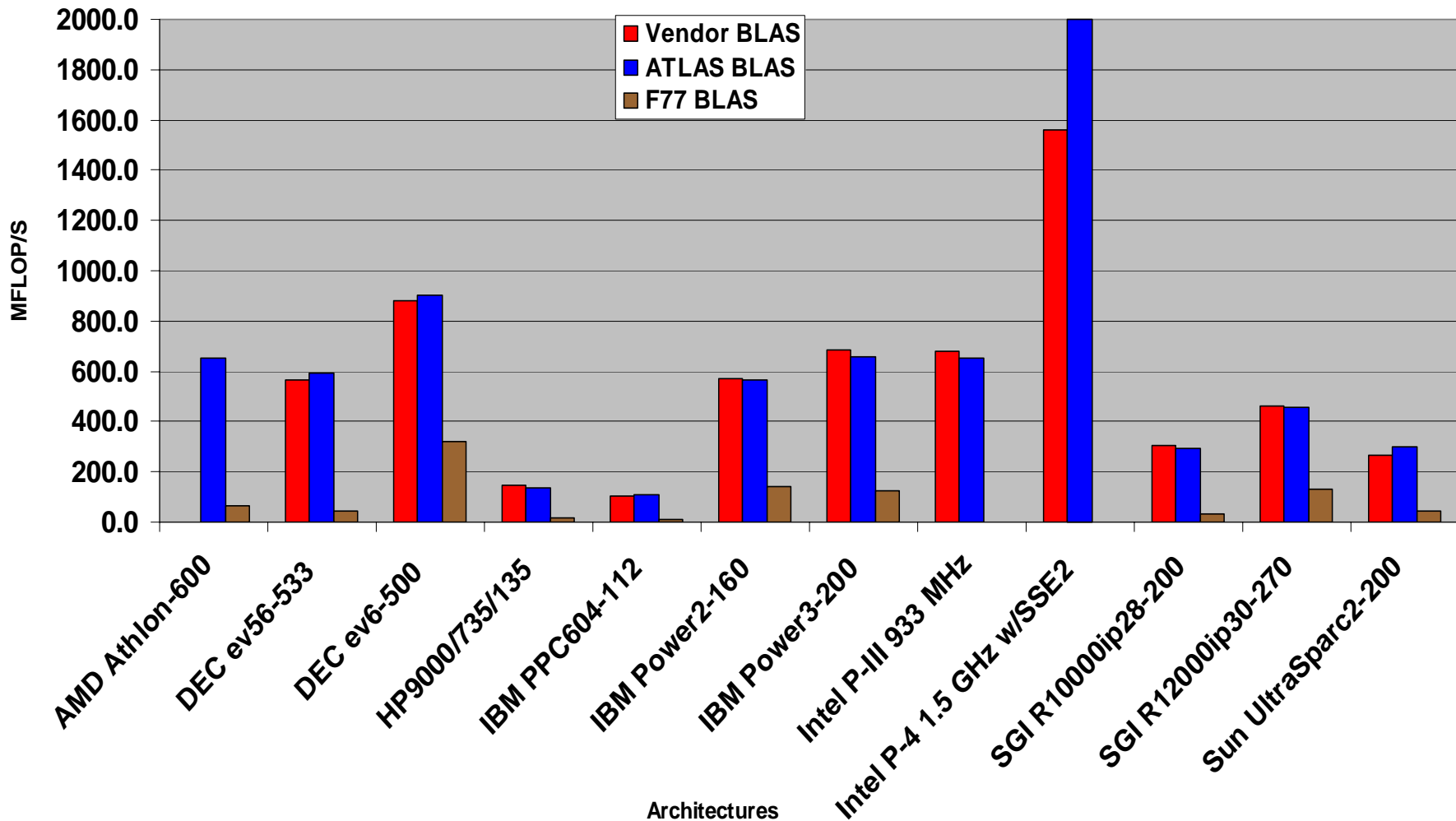
Software Generation Strategy - ATLAS BLAS

- Parameter study of the hw
- Generate multiple versions of code, w/difference values of key performance parameters
- Run and measure the performance for various versions
- Pick best and generate library
- Level 1 cache multiply optimizes for:
 - TLB access
 - L1 cache reuse
 - FP unit usage
 - Memory fetch
 - Register reuse
 - Loop overhead minimization



- Takes ~ 20 minutes to run, generates Level 1, 2, & 3 BLAS
- “New” model of high performance programming where critical code is machine generated using parameter optimization.
- Designed for RISC arch
 - Super Scalar
 - Need reasonable C compiler
- Today ATLAS is used within various ASCI and SciDAC activities and by Matlab, Mathematica, Octave, Maple, Debian, Scyld Beowulf, SuSE,...
 - Eg: 725 Mflop/s on this laptop

ATLAS (DGEMM $n = 500$)



- ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.

Reformulating/Rearranging/Reuse

- Example is the reduction to narrow band from for the SVD

$$A_{new} = A - uy^T - wv^T$$

$$y_{new} = A^T u$$

$$w_{new} = A_{new} v$$

- Fetch each entry of A once
- Restructure and combined operations
- Results in a speedup of $> 30\%$

Solving Large Sparse Non-Symmetric Systems of Linear Equations Using BiCG-Stab

```
Compute  $r^{(0)} = b - Ax^{(0)}$  for some initial guess  $x^{(0)}$ .  
Choose  $\tilde{r}^{(0)}$  (for example,  $\tilde{r}^{(0)} = r^{(0)}$ ).  
for  $i = 1, 2, \dots$   
  solve  $Mx^{(i-1)} = r^{(i-1)}$   
  solve  $M^T \tilde{x}^{(i-1)} = \tilde{r}^{(i-1)}$   
   $\rho_{i-1} = x^{(i-1)T} \tilde{r}^{(i-1)}$   
  if  $\rho_{i-1} = 0$ , method fails  
  if  $i = 1$   
     $p^{(i)} = x^{(i-1)}$   
     $\tilde{p}^{(i)} = \tilde{x}^{(i-1)}$   
  else  
     $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$   
     $p^{(i)} = x^{(i-1)} + \beta_{i-1} p^{(i-1)}$   
     $\tilde{p}^{(i)} = \tilde{x}^{(i-1)} + \beta_{i-1} \tilde{p}^{(i-1)}$   
  endif  
   $q^{(i)} = Ap^{(i)}$   
   $\tilde{q}^{(i)} = A^T \tilde{p}^{(i)}$   
   $\alpha_i = \rho_{i-1} / \tilde{p}^{(i)T} q^{(i)}$   
   $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$   
   $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$   
   $\tilde{r}^{(i)} = \tilde{r}^{(i-1)} - \alpha_i \tilde{q}^{(i)}$   
  check convergence; continue if necessary  
end
```

- Example of optimizations
 - Combining 2 vector ops into 1 loop
 - Simplifying indexing
 - Removal of “if test” within loop

Optimization of BiCG-Stab

10% - 20% Improvement

Matrix	PIII no	PIII Opt	P4 no	P4 Opt	Itanium no	Itanium Opt
jpwh_991	0.015	0.01	0.007	0.007	0.031	0.026
memplus	12.004	8.47	2.546	1.976	13.606	12.334
orsreg_1	0.574	0.525	0.132	0.125	0.561	0.479
psmigr_1	5.297	4.9	0.911	0.94	8.181	7.78
raefsky3	237.819	154.077	43.005	31.187	297.044	283.738
saylr4	11.388	12.126	2.478	2.708	10.272	8.754
sherman3	1.992	1.991	0.473	0.469	1.677	1.439
sherman5	0.333	0.366	0.083	0.085	0.3	0.257
venkat01	19.877	15.536	3.407	3.306	28.24	26.891
wang3	6.634	6.057	1.085	1.243	7.698	7.195
wang4	6.781	5.246	1.19	1.123	7.684	6.991

Execution time in seconds

CG Variants by Dynamic Selection at Run Time

- Variants combine inner products to reduce communication bottleneck at the expense of more scalar ops.
- Same number of iterations, no advantage on a sequential processor
- With a large number of processor and a high-latency network may be advantages.
- Improvements can range from 15% to 50% depending on size.

Classical

Norm calculation:
 $\text{error} = \sqrt{r^t r}$

Preconditioner application:
 $z \leftarrow M^{-1} r$

Matrix-vector product:

Inner products 1:

$\rho \leftarrow z^t r$

$\beta \leftarrow \rho / \rho_{\text{old}}$

Search direction update:
 $p \leftarrow z + \beta p$

Matrix-vector product:
 $ap \leftarrow A \times p$

Preconditioner application:

Inner products 2:

$\pi \leftarrow p^t ap$

$\alpha = \rho / \pi$

Residual update:
 $r \leftarrow r - \alpha Ap$

3 separate inner products

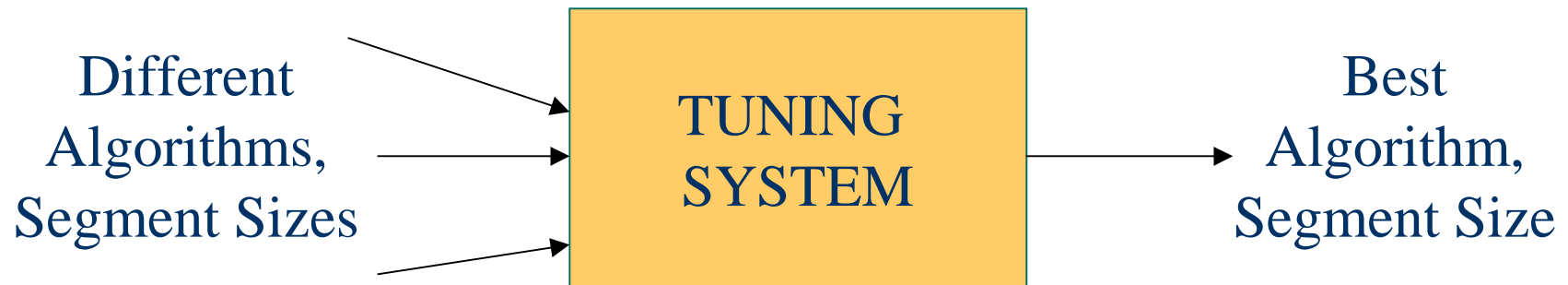
CG Variants by Dynamic Selection at Run Time

- Variants combine inner products to reduce communication bottleneck at the expense of more scalar ops.
- Same number of iterations, no advantage on a sequential processor
- With a large number of processor and a high-latency network may be advantages.
- Improvements can range from 15% to 50% depending on size.

Classical	Saad/Meurant	Chronopoulos/Gear	Eijkhout
<i>Norm calculation:</i>			
$\text{error} = \sqrt{r^t r}$			
<i>Preconditioner application:</i>			
$z \leftarrow M^{-1}r$	$z \leftarrow z - \alpha q$	$z \leftarrow M^{-1}r$	id
<i>Matrix-vector product:</i>			
		$az \leftarrow A \times z$	id
<i>Inner products 1:</i>			
$\rho \leftarrow z^t r$	$\rho_{\text{predict}} \leftarrow -\rho_{\text{true}} + \alpha^2 \mu$	$\text{error} = \sqrt{r^t r}$ $\rho \leftarrow z^t r$ $\zeta \leftarrow z^t az$	$\text{error} = \sqrt{r^t r}$ $\rho \leftarrow z^t r$ $\zeta \leftarrow z^t az$ $\epsilon \leftarrow (M^{-1}r)^t (Ap)$
$\beta \leftarrow \rho / \rho_{\text{old}}$	$\beta = \rho_{\text{predict}} / \rho_{\text{old}}$	$\beta \leftarrow \rho / \rho_{\text{old}}$	id
<i>Search direction update:</i>			
$p \leftarrow z + \beta p$	id	id	id
<i>Matrix-vector product:</i>			
$ap \leftarrow A \times p$	id	$ap \leftarrow az + \beta ap$	id
<i>Preconditioner application:</i>			
	$q \leftarrow M^{-1}ap$		
<i>Inner products 2:</i>			
$\pi \leftarrow p^t ap$	$\pi \leftarrow p^t ap$ $\mu \leftarrow ap^t q$ $\text{error} = \sqrt{r^t r}$ $\rho_{\text{true}} = z^t r$	$\pi \leftarrow \zeta - \beta^2 \pi$	$\pi \leftarrow \zeta + \beta \epsilon$
$\alpha = \rho / \pi$	$\dots \rho_{\text{true}} \dots$	$\alpha = \rho / \pi$	
<i>Residual update:</i>			
$r \leftarrow r - \alpha Ap$	id	id	id
3 separate inner products	4 combined 1 extra vector update	3 combined id	4 combined id

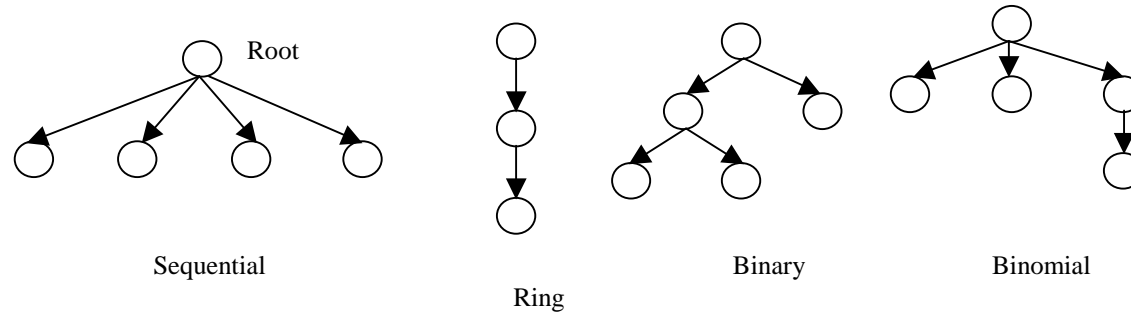
Machine-Assisted Application Development and Adaptation

- **Communication libraries**
 - Optimize for the specifics of one's configuration.
 - A specific MPI collective communication algorithm may not give best results on all platforms.
 - Choose collective communication parameters that give best results for the system.
- **Algorithm layout and implementation**
 - Look at the different ways to express implementation



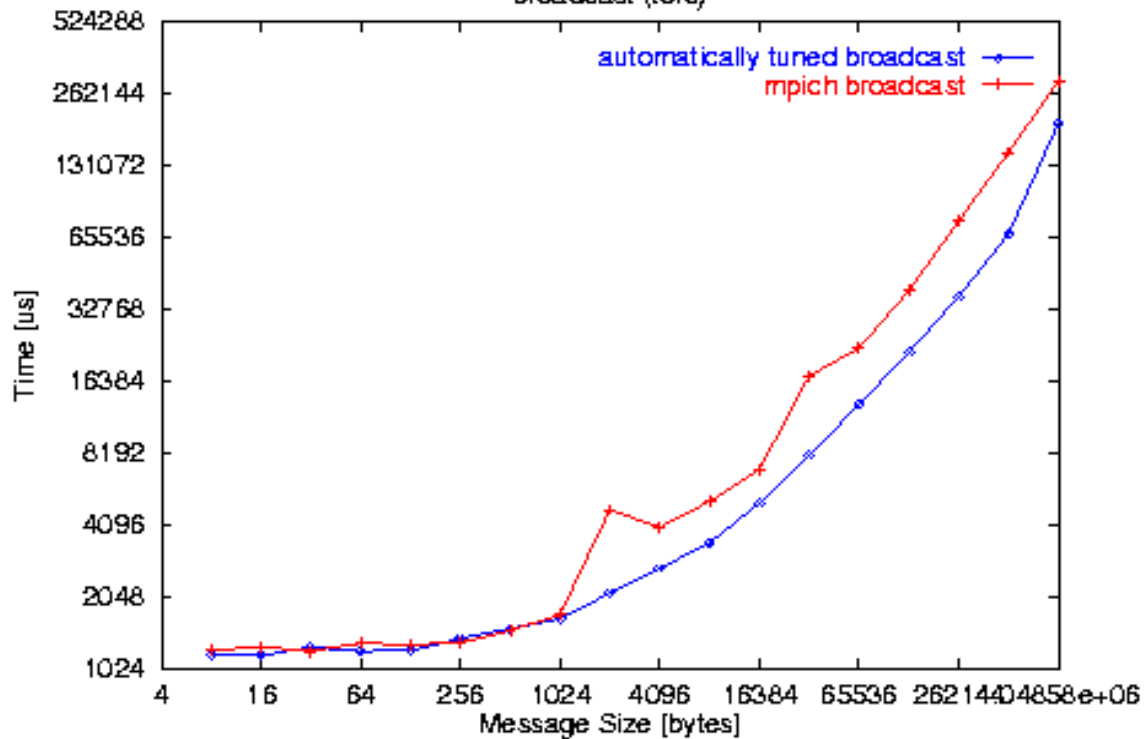
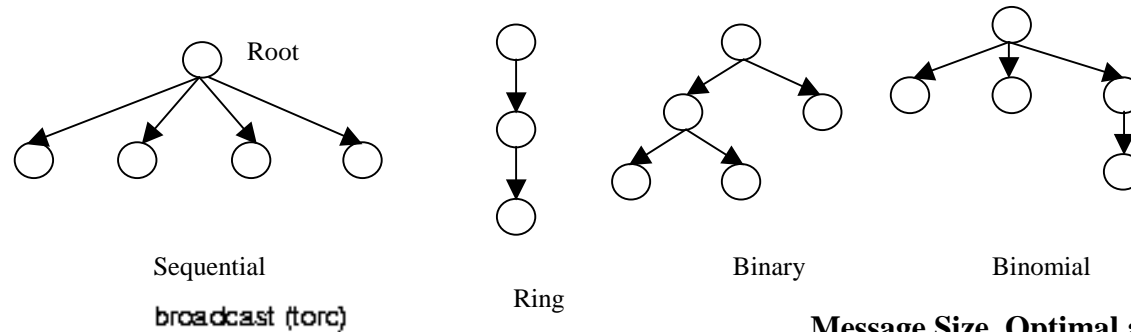
Work in Progress: SANS Approach Applied to Broadcast

(PII 8 Way Cluster with 100 Mb/s switched network)



Work in Progress: SANS Approach Applied to Broadcast

(PII 8 Way Cluster with 100 Mb/s switched network)

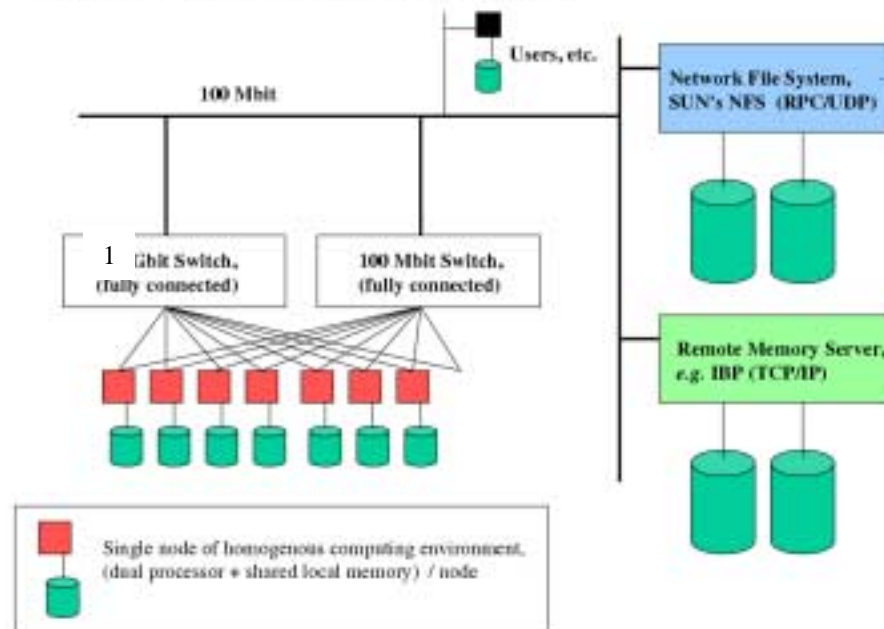


Message Size (bytes)	Optimal algorithm	Buffer Size (bytes)
8	binomial	8
16	binomial	16
32	binary	32
64	binomial	64
128	binomial	128
256	binomial	256
512	binomial	512
1K	sequential	1K
2K	binary	2K
4K	binary	2K
8K	binary	2K
16K	binary	4K
32K	binary	4K
64K	ring	4K
128K	ring	4K
256K	ring	4K
512K	ring	4K
1M	binary	4K

LAPACK For Clusters

- Developing middleware which couples cluster system information with the specifics of a user problem to launch cluster based applications on the “best” set of resource available.

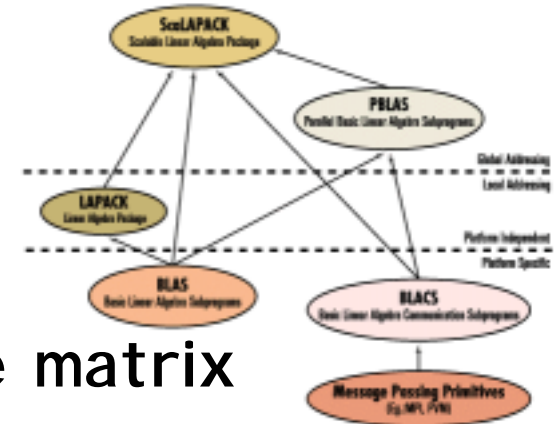
Sample computing environment...



- Using ScaLAPACK as the prototype software

ScaLAPACK

ScaLAPACK
A Software Library for Linear Algebra Computations on Distributed-Memory



- ScaLAPACK is a portable distributed memory numerical library
- Complete numerical library for dense matrix computations
- Designed for distributed parallel computing (MPP & Clusters) using MPI
- One of the first math software packages to do this
- Numerical software that will work on a heterogeneous platform
- In use today by various ASCI and SciDAC efforts, IBM, HP-Convex, Fujitsu, NEC, Sun, SGI, Cray, NAG, IMSL, ...
 - Tailor performance & provide support

How ScaLAPACK Works

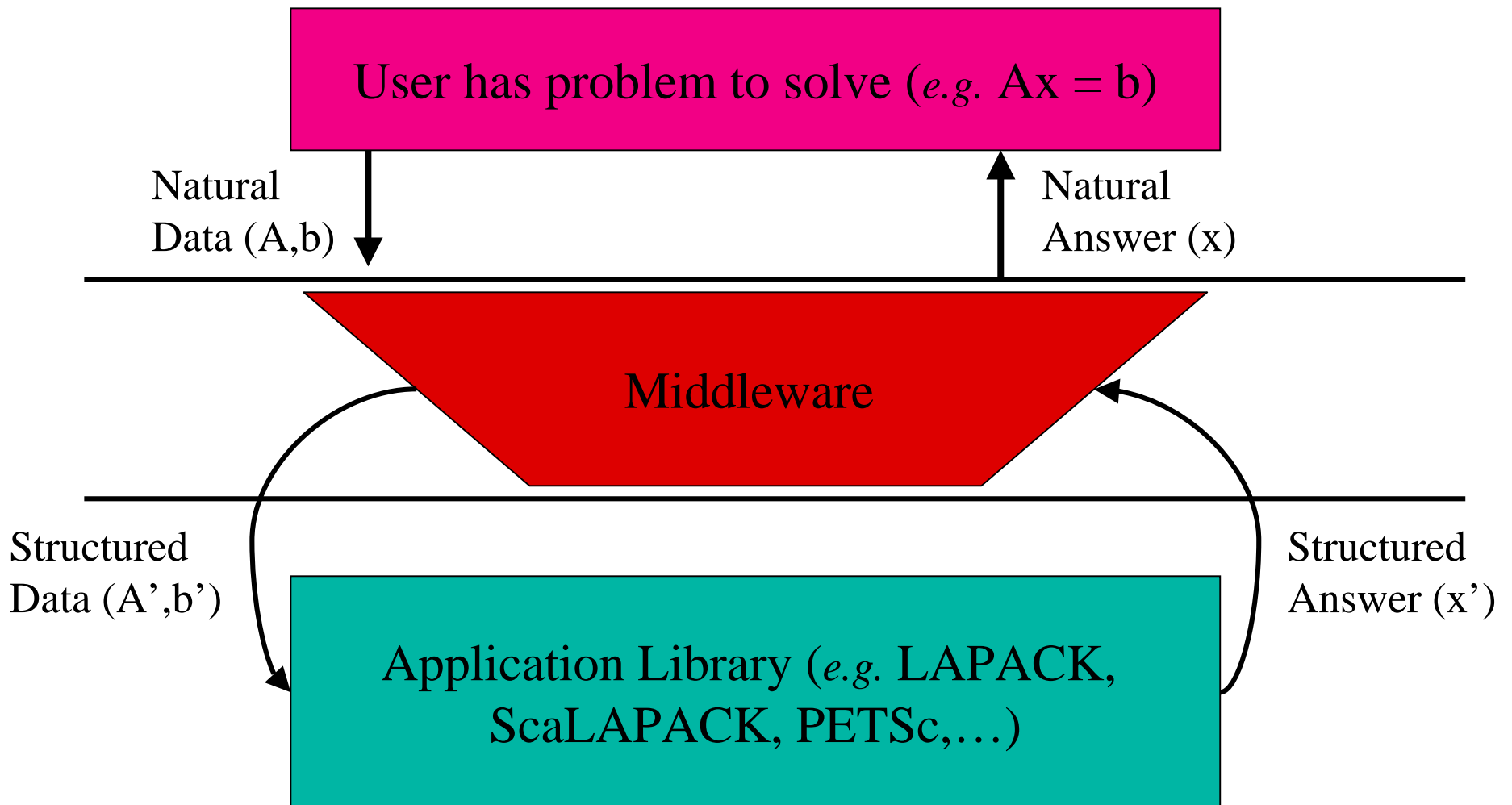
- To use ScaLAPACK a user must:
 - Download the package and auxiliary packages (like PBLAS, BLAS, BLACS, & MPI) to the machines.
 - If heterogeneous collection of machines, make sure proper versions available.
 - Write a SPMD program which
 - Sets up the logical 2-D process grid
 - Places the data on the logical process grid
 - Calls the library routine in a SPMD fashion
 - Collects the solution after the library routine finishes
 - The user must allocate the processors and decide the number of processes the application will run on
 - The user must commit to a certain # of processors then start the application
 - "mpirun -np *N* user_app"
 - Upon completion, return the processors to the pool of resources

Note: the number of processors is fixed by the user before the run

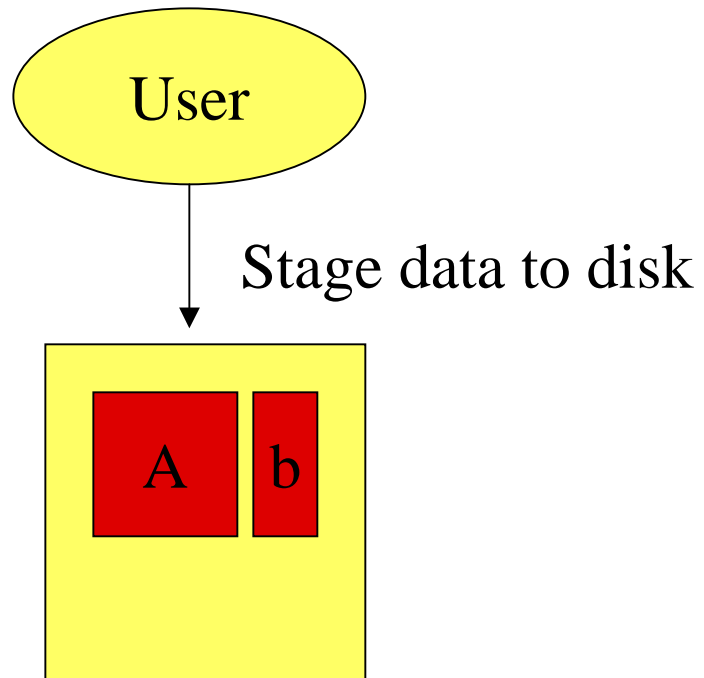
LAPACK For Clusters

- Idea to make it easy to use your cluster to solve dense matrix problems.
- As simple as a conventional call to LAPACK
- Make decisions on which machines to use based on the user's problem and the state of the system
 - Determinate machines that can be used
 - Optimize for the best time to solution
 - Distribute the data on the processors and collections of results
 - Start the SPMD library routine on all the platforms

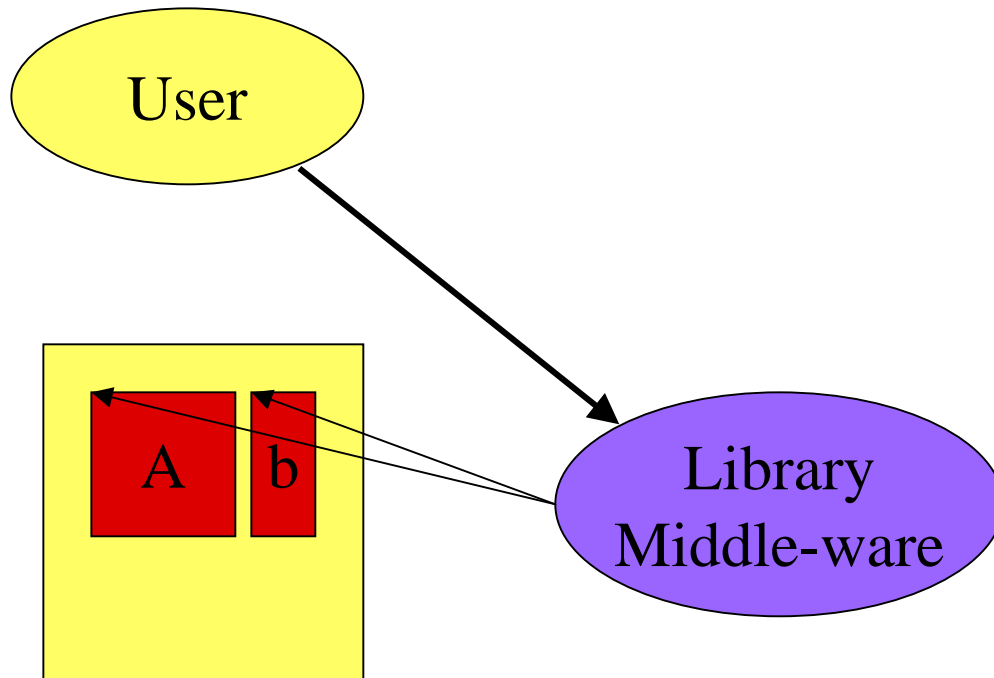
Big Picture...



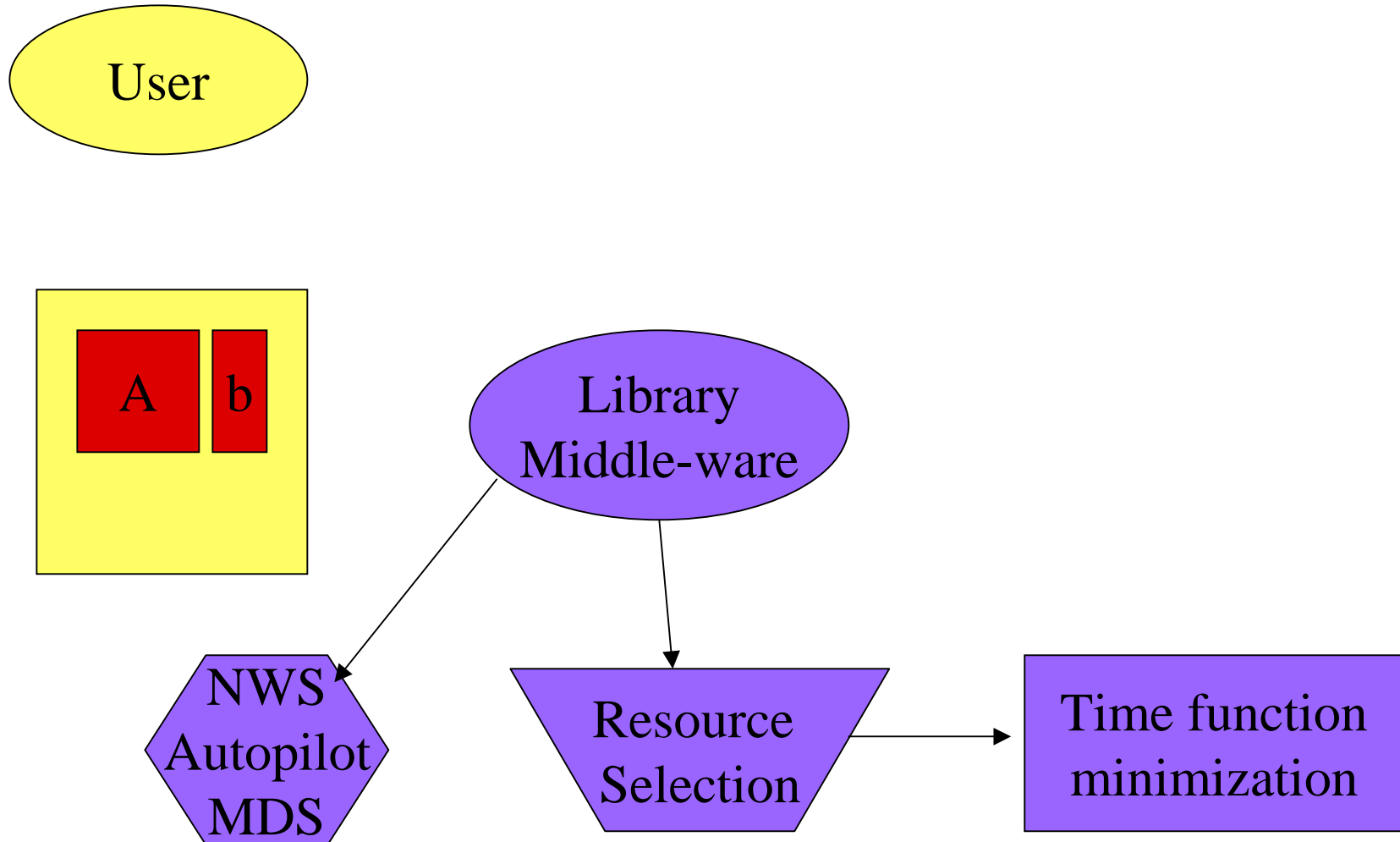
File System -based



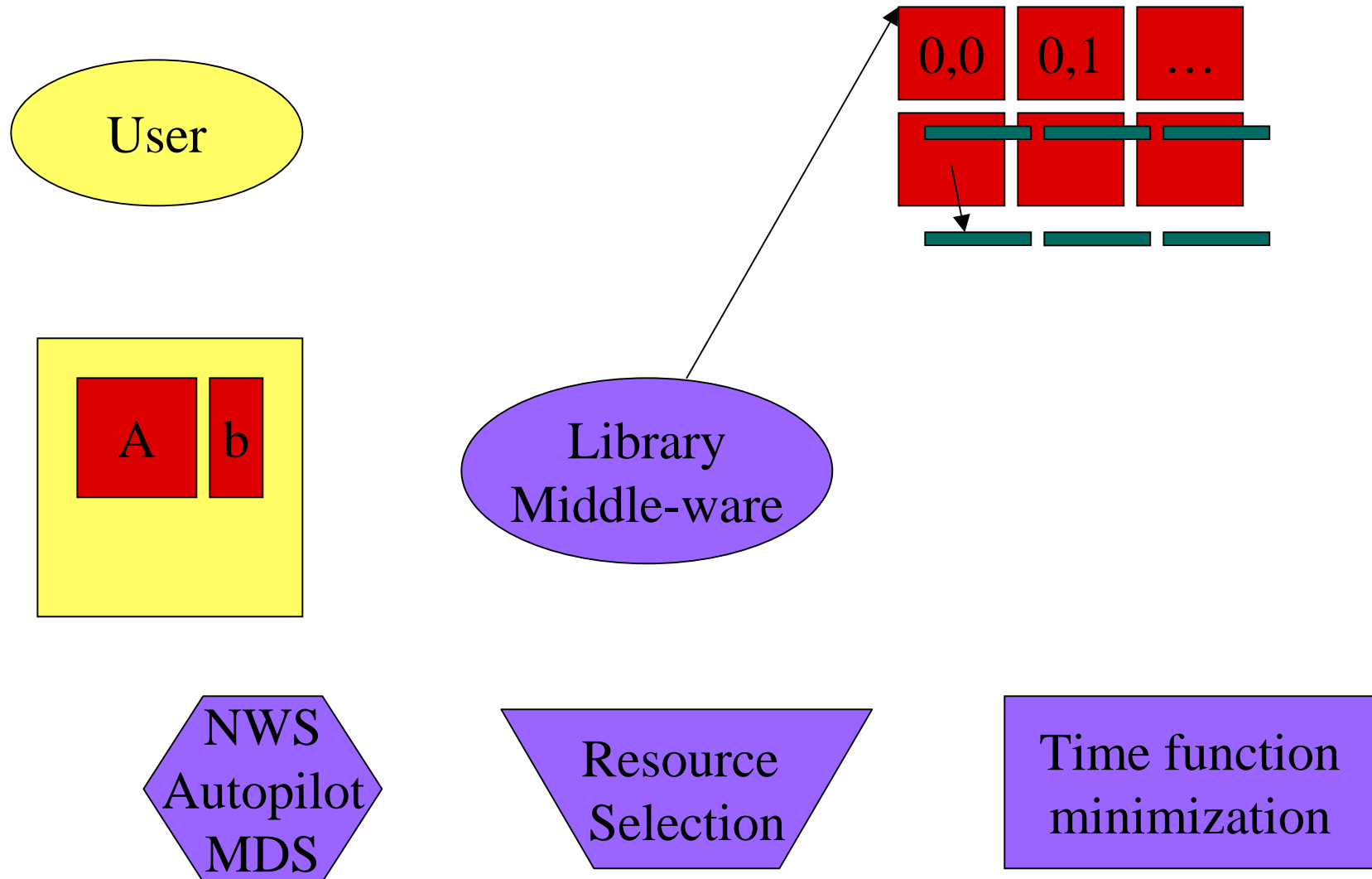
File System -based



File System -based



File System -based



Can use Grid infrastructure, i.e. Globus/NWS, but doesn't have to.

Resource Selector

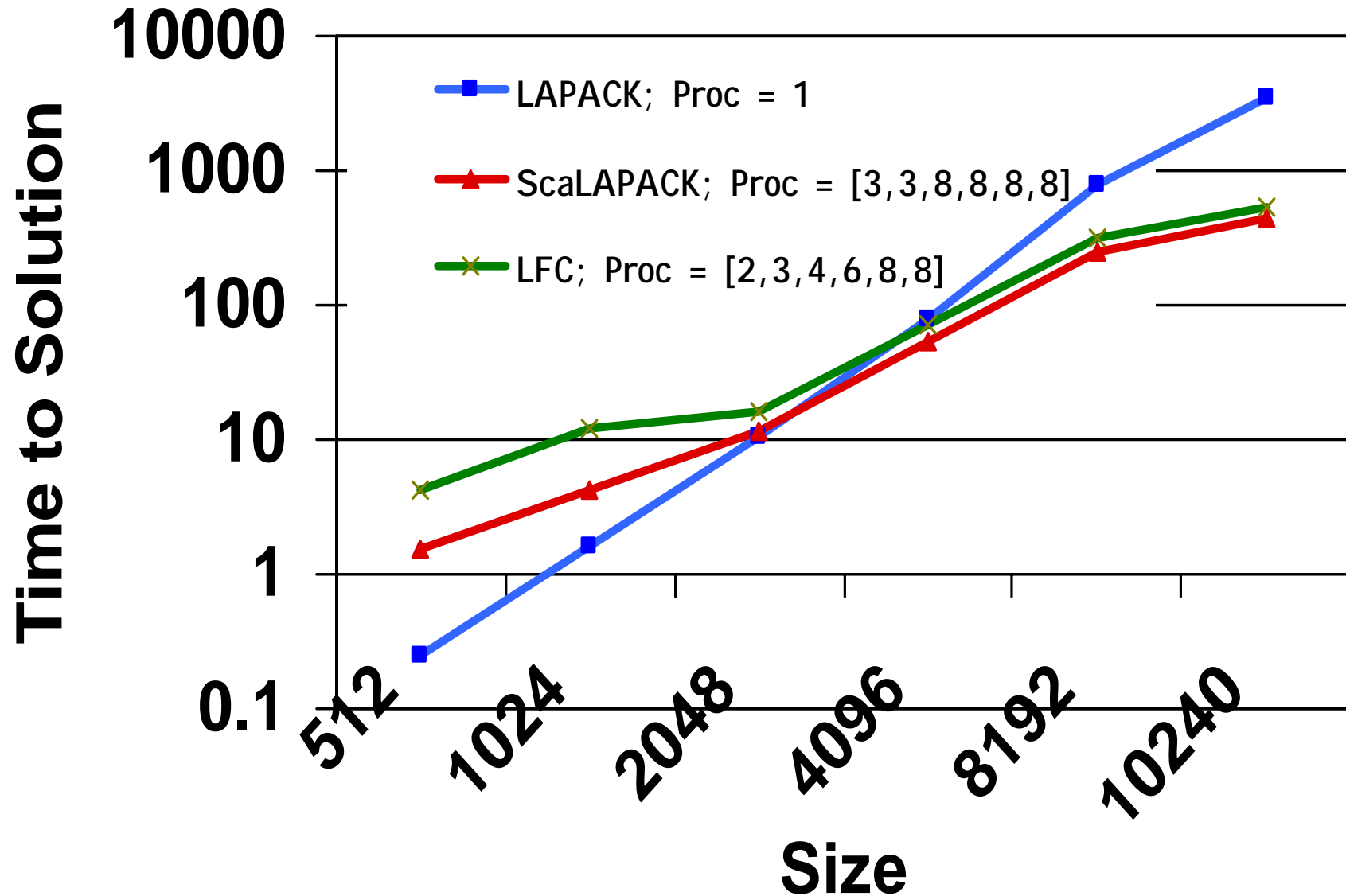


- Uses Rich Wolski's (UCSB) NWS to build an array of values for the machines that are available for the user.
 - 2 matrices (bw,lat) 3 arrays (load, cpu, memory available)
- Generated dynamically by library routine

Bandwidth				Latency				Load	Memory	CPU Performance
X	X	∴	X	X	X	∴	X	X	X	X
X	X	∴	X	X	X	∴	X	X	X	X
∴	∴	∴	∴	∴	∴	∴	∴
X	X	∴	x	X	X	∴	x	X	X	X

$$Ax = b$$

Cluster of 8 Pentium III 933 MHz



LAPACK For Clusters (LFC)

- LFC will automate much of the decisions in the Cluster environment to provide best time to solution.
 - Adaptivity to the dynamic environment.
 - As the complexities of the Clusters and Grid increase need to develop strategies for self adaptability.
 - Handcrafted developed leading to an automated design.
- Developing a basic infrastructure for computational science applications and software in the Cluster and Grid environment.
 - Lack of tools is hampering development today.
- Plan to do suite: LU, Cholesky, QR, Symmetric eigenvalue, and Nonsymmetric eigenvalue
- Model for more general framework

Tools for Performance Evaluation



- Timing and performance evaluation has been an art
 - Resolution of the clock
 - Issues about cache effects
 - Different systems
 - Can be cumbersome and inefficient with traditional tools
- Situation about to change
 - Today's processors have internal counters



Performance Counters

- Almost all high performance processors include hardware performance counters.
- Some are easy to access, others not available to users.
- On most platforms the APIs, if they exist, are not appropriate for the end user or well documented.
- Existing performance counter APIs
 - Compaq Alpha EV 6 & 6/7
 - SGI MIPS R10000
 - IBM Power Series
 - CRAY T3E
 - Sun Solaris
 - Pentium Linux and Windows
 - IA-64
 - HP-PA RISC
 - Hitachi
 - Fujitsu
 - NEC





Performance Data That May Be Available

- Cycle count
- Floating point instruction count
- Integer instruction count
- Instruction count
- Load/store count
- Branch taken / not taken count
- Branch mispredictions
- Pipeline stalls due to memory subsystem
- Pipeline stalls due to resource conflicts
- I/D cache misses for different levels
- Cache invalidations
- TLB misses
- TLB invalidations

PAPI - Supported Processors

- Intel Pentium, II, III, Itanium, (P 4 in alpha testing now)
 - Linux 2.4, 2.2, 2.0 and perf kernel patch
- IBM Power 3, 604, 604e (Power 4 coming)
 - For AIX 4.3 and pmtoolkit (in 4.3.4 available)
 - (laderose@us.ibm.com)
- Sun UltraSparc I, II, & III
 - Solaris 2.8
- SGI IRIX/MIPS
- AMD Athlon
 - Linux 2.4 and perf kernel patch
- Cray T3E, SV1, SV2
- Windows 2K and XP
- To download software see:

<http://icl.cs.utk.edu/papi/>

Work in progress on Compaq Alpha
Fortran, C, and MATLAB bindings



Early Users of PAPI



- DEEP/PAPI (Pacific Sierra)
http://www.psrv.com/deep_papi_top.html
- TAU (Allen Mallony, U of Oregon)
<http://www.cs.uoregon.edu/research/paracomp/tau/>
- SvPablo (Dan Reed, U of Illinois)
<http://vibes.cs.uiuc.edu/Software/SvPablo/svPablo.htm>
- Cactus (Ed Seidel, Max Plank/U of Illinois)
<http://www.aei-potsdam.mpg.de>
- Vprof (Curtis Janssen, Sandia Livermore Lab)
<http://aros.ca.sandia.gov/~cljanss/perf/vprof/>
- Cluster Tools (Al Geist, ORNL)
- DynaProf



What is DynaProf?

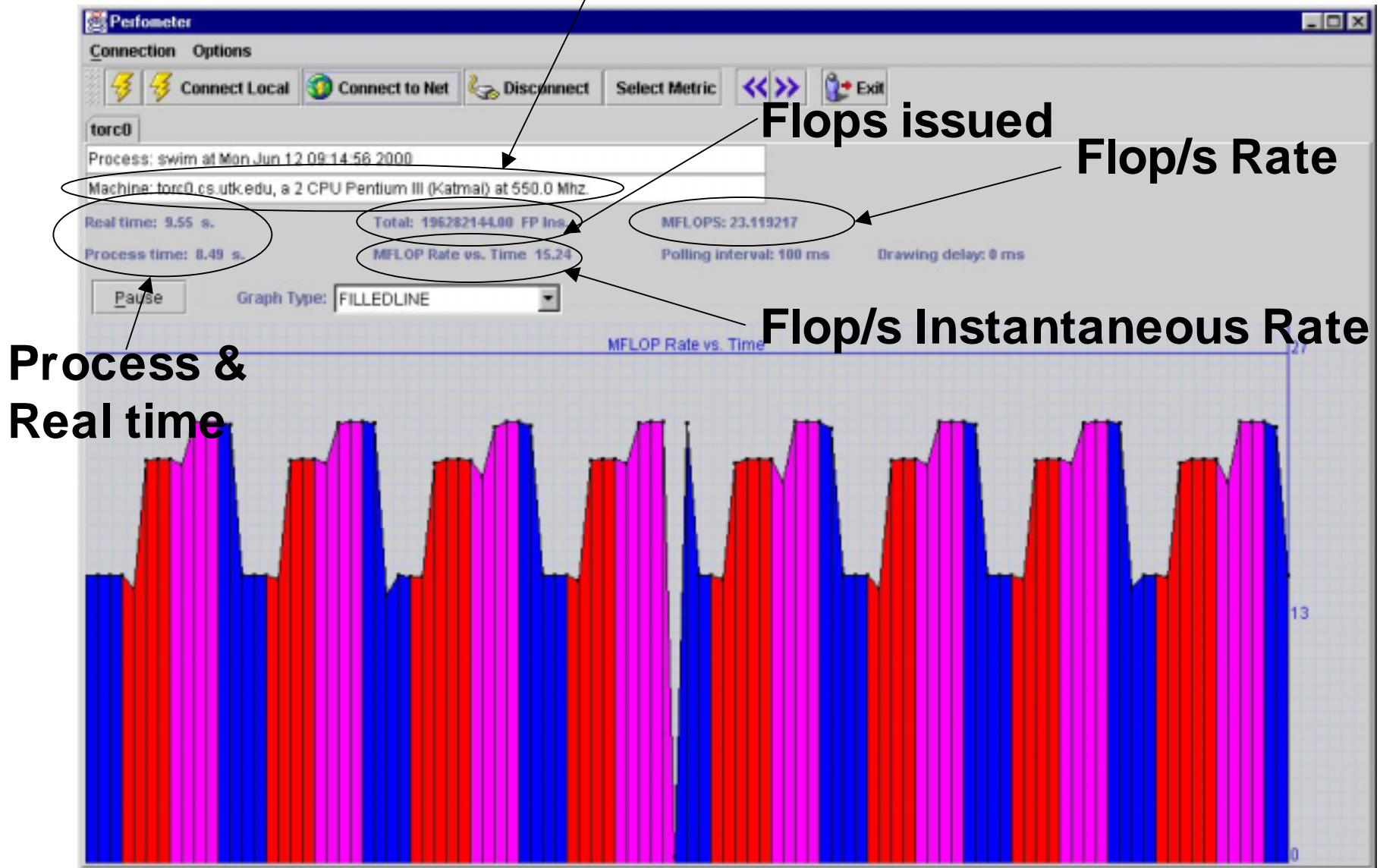
- A portable tool to dynamically instrument a running executable with *Probes* that monitor application performance.
- Simple command line interface.
- Java based GUI interface.
- Open Source Software.
- Built on and in collaboration with Bart Miller and Jeff Hollingsworth Paradyn project at U. Wisconsin and Dyninst project at U. Maryland
 - <http://www.paradyn.org/>
 - <http://www.dyninst.org/>
- A work in progress...

Dynamic Instrumentation:

- Operates on a running executable.
- Identifies instrumentation *points* where code can be inserted.
- Inserts code *snippets* at selected *points*.
- *Snippets* can collect and monitor performance information.
- *Snippets* can be removed and reinserted dynamically.
- Source code not required, just executable

Perfometer/ DynaProf

Machine info



Futures for Numerical Algorithms and Software on Clusters and Grids

- Retargetable Libraries - Numerical software will be adaptive, exploratory, and intelligent
- Determinism in numerical computing will be gone.
 - After all, its not reasonable to ask for exactness in numerical computations.
 - Auditability of the computation, reproducibility at a cost
- Importance of floating point arithmetic will be undiminished.
 - 16, 32, 64, 128 bits and beyond.
- Reproducibility, fault tolerance, and auditability
- Adaptivity is a key so applications can effectively use the resources.

Collaborators

- **ATLAS**
 - Antoine Petitet, Sun
 - Clint Whaley, FSU
- **Sparse Ops**
 - Victor Eijkhout, UTK
- **Optimizing communication**
 - Sathish Vadhiyar, UTK
- **LFC**
 - Jeffrey Chen, UTK
 - Piotr Luszczek, UTK
 - Kenny Roche, UTK
- **PAPI**
 - Kevin London, UTK
 - Shirley Moore, UTK
 - Phil Mucci, UTK
- **DynaProf**
 - Jeff Hollingsworth, UMaryland
 - Bart Miller, UWisconsin
 - Dan Terpstra, UTK
 - Haihang You, UTK
- **Software Availability**
 - PAPI
 - <http://icl.cs.utk.edu/papi/>
 - ATLAS
 - <http://icl.cs.utk.edu/atlas/>
 - LFC
 - 5 drivers from ScaLAPACK by the end of summer
 - DynaProf
 - tarball coming soon
- **Earth Simulator slides at**
 - <http://www.cs.utk.edu/~dongarra/esc.pdf>